



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/912,011	07/24/2001	Gerald D. Zuraski JR.	5500-67400	4622

7590

06/03/2004

Rory D. Rankin
Conley, Rose & Tayon, P.C.
P.O. Box 398
Austin, TX 78767

EXAMINER

RAVINDRAN, LATHA

ART UNIT

PAPER NUMBER

2183

DATE MAILED: 06/03/2004

8

Please find below and/or attached an Office communication concerning this application or proceeding.

3

Office Action Summary	Application No. 09/912,011	Applicant(s) ZURASKI ET AL.	
	Examiner Latha Ravindran	Art Unit 2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 24 July 2001.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-20 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-20 is/are rejected.
- 7) ☒ Claim(s) 1-20 is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☒ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 5 August 2002 is/are: a) ☐ accepted or b) ☒ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date <u>4.7</u> . | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. Claims 1 – 20 have been examined. Claims 1 – 20 have been rejected.
2. Receipt is acknowledged for Drawings, received 10/15/2001, IDS, received 4/29/2002, Request to Rescind Previous Non-Publication Request, received 7/22/2002, Drawings, received 8/5/2002, and IDS, received 2/20/2003.

Drawings

3. Figures 1,2,3,4,19,24 26, and any other pertinent drawings should be designated by a legend such as --Prior Art-- because only that which is old is illustrated. See MPEP § 608.02(g).
4. A proposed drawing correction or corrected drawings are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

Specification

5. Applicant is reminded of the proper language and format for an abstract of the disclosure.

The abstract should be in narrative form and generally limited to a single paragraph on a separate sheet within the range of 50 to 150 words. It is important that the abstract not exceed 150 words in length since the space provided for the abstract on the computer tape used by the printer is limited. The abstract should describe the disclosure sufficiently to assist readers in deciding whether there is a need for consulting the full patent text for details.

The language should be clear and concise and should not repeat information given in the title.

6. The abstract is objected under 37 CFR 1.72(b) because it exceeds 150 words.
7. The first sentence of the abstract on line 1 reiterates information given in the title. Information given in the title should not be repeated. See MPEP 608.01(b).

8. Correction of the abstract is required.
9. The disclosure is objected to because of the following informalities:
 - The section headings are not in proper format. See MPEP 608.01(a).Appropriate correction is required.
10. The lengthy specification has not been checked to the extent necessary to determine the presence of all possible minor errors. Applicant's cooperation is requested in correcting any errors of which applicant may become aware in the specification.

Claim Objections

11. Claims 1 – 20 are objected to because of the following informalities:

Claim 1,9,17:

12. Claim 1 states,
line 2 "...detecting a first level cache does not contain *branch prediction information*.."
line 4 "...determining whether a second level cache contains *branch prediction information*..."

The term "branch prediction information" is ambiguous in line 4.

13. Claim 9 states,
line 3 "a first level cache configured to store *branch prediction information*;..."
line 4 "...a second level cache configured to store *branch prediction information*..."

The term "branch prediction information" is ambiguous in line 4.

14. Claim 17 states,
line 6 "...a second level cache configured to store *branch prediction information*..."

Art Unit: 2183

line 8 "...detect said first level cache does not contain *branch prediction information...*"

The term "branch prediction information" is ambiguous in line 8.

15. If "branch prediction information" in the following line (lines 4,4, and 8 above in claim 1,9, and 17 respectively) is meant to be different from the term "branch prediction information" stated in the preceding line (lines 2,3,and 6 above in claim 1,9, and 17 respectively), then the term in the following line should be renamed to prevent any confusion. Otherwise, the word "said" should come before the term in the following line to clarify that they are the same.

16. The examiner notes that for further examination, branch prediction information is interpreted to mean the same thing, which is further defined below in the claim rejections..

Claim 5,13:

17. Claim 5 states, "The method of claim 4, wherein said branch instruction is fetched from said second level cache." Said second level cache refers to the second level cache containing branch prediction information. (Claim 1) See Fig. 2, Element 260 and Fig. 20, Element 260. However, in the specification, the branch instruction is fetched from the second level instruction cache. (Page 39, Line 18 – 20) Refer to Figure 20, Elements 1828 and 2104.

18. Claim 13 states, "The mechanism of claim 12, wherein said branch instruction is fetched from said second level cache." Said second level cache refers to the second level cache containing branch prediction information. (Claim 9) See Fig. 2, Element 260

and Fig. 20, Element 260. However, in the specification, the branch instruction is fetched from the second level instruction cache. (Page 39, Line 18 – 20) Refer to Figure 20, Elements 1828 and 2104.

19. The examiner notes that for further examination, the said second level cache in claim 5 and claim 13 will be interpreted as the second level instruction cache (Figure 20, Element 1828). Appropriate correction of the claims is required.

20. Claims not specifically mentioned, but listed as objected above, are objected based on their dependency to the above claims.

Claim Rejections - 35 USC § 102

21. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

22. Claims 1,2,9, and 10 are rejected under 35 U.S.C. 102(b) as being anticipated by Perleberg et al. in *Branch Target Buffer Design and Optimization*.

Claim 1:

23. A method comprising:

- detecting a first level cache does not contain branch prediction information corresponding to a first address; (Page 409, first column, first,second, and fourth paragraphs, “....Branches in lower performance levels are moved to the highest level on a branch taken execution;...”, page 398, first column, fourth paragraph, “The Branch Target Buffer (BTB) can be used to reduce the performance penalty

of branches by predicting the path of the branch and caching information about the branch [30]...As each instruction is fetched from memory, the instruction address is used to index into the BTB. If a valid BTB entry is found for that address, that instruction is a branch"/ The first level cache is a first level branch target buffer (BTB), with the same functionality as a BTB. The branch prediction information is the cached information about the branch stored in the BTB. The instruction address is the first address. The case exists where the branch target buffer does not have a valid BTB entry for an instruction address. As a result, the case exists where the branch target buffer does not contain branch prediction information corresponding to the instruction address. This scenario is detected by a multilevel BTB as the prediction information could come from other BTBs, namely lower levels.)

- determining whether a second level cache contains branch prediction information corresponding to said first address; (Page 396, second column, third paragraph, "It is also possible to create a multilevel BTB, with different designs at each level, each containing a different subset of these items.", Page 409, first column, fourth paragraph, "Branches in lower performance levels are moved to the highest level on a branch taken execution; ...", Page 398, first column, fourth paragraph, "The Branch Target Buffer (BTB) can be used to reduce the performance penalty of branches by predicting the path of the branch and caching information about the branch [30]...As each instruction is fetched from memory, the instruction address is used to index into the BTB. If a valid BTB entry is found for that address, the

instruction is a branch.”/ The second level cache is a second level BTB, with the same functionality as a BTB. The branch prediction information is the cached information about the branch stored in the BTB. The instruction address is the first address. The case exists where the branch target buffer has a valid BTB entry for an instruction address since a valid BTB entry is “found”. It is determined to have the branch prediction information about the instruction address.)

- rebuilding a first branch prediction using said information in response to determining said second level cache contains said information (Page 409, first column, fourth paragraph, “Branches in lower performance levels are moved to the highest level on a branch taken execution; this has the benefit that the target instructions can be fetched without increasing memory demands.”/The first branch prediction is the branch prediction information stored in the first level BTB. After branches stored the second level BTB have been predicted correctly and executed as taken, they are stored in the first level BTB. This would entail filling in the entry of the first level BTB, or first branch prediction, with the appropriate information.)
- wherein said information comprises a subset of said first branch prediction (Page 410, See Table XV/ The second level BTBs in BTB 6 and 7 contain a subset of information from their respective first level BTBs); and
- storing said first branch prediction in a first entry of said first level cache, (Page 409, first column, fourth paragraph, “Branches in lower performance levels are

moved to the highest level on a branch taken execution; this has the benefit that the target instructions can be fetched without increasing memory demands. / A first entry is an entry of the BTB.)

- wherein said first entry corresponds to said first address. (Page 398, first column fourth paragraph, "As each instruction is fetched from memory, the instruction address is used to index into the BTB." The branch prediction entries are stored in the first level cache according to their instruction address)

Claim 2:

24. The method of claim 1, further comprising:

- determining if said first entry of said first level cache is available; (Page 398, first column, fourth paragraph, "If a valid BTB entry is found for that address, the instruction is a branch." / Therefore, the entry can be invalid and, thus, available)
- evicting contents of said first entry (Page 409, first column, fourth paragraph, "Entries at higher performance levels are moved to lower performance levels one level at a time as they are replaced.")
- in response to detecting said first entry is not available; (Page 400, column 1, paragraph 2, "An optimization problem exists if a maximum performance BTB is to be designed with a limited number of bits. As both more entries and more information per entry increase performance, the problem is selecting the number of entries and the amount of information per entry that will maximize performance. "Page 403, column 2, fourth paragraph, "The second concept presented in the *BTB Design Optimization* section is, "When it is necessary to

discard a branch from the BTB, discard the branch with the minimum performance potential. The Least Recently Used (LRU) algorithm is known by experience to work well for replacing (discarding) entries in memory systems, such as main memory pages and cache lines."/ Because of space limitations, the case exists where the entry in the first level cache is not available because it is being used.) and

- storing a subset of said contents in said second level cache responsive to said eviction. (Page 409, column 1 and 2, fourth paragraph, "Entries at higher performance levels are moved to lower performance levels one level at a time as they are replaced. " Page 410, See Table XV. BTB 6 7)

Claim 9:

25. A branch prediction mechanism comprising:

- a first level cache configured to store branch prediction information (Page 409, first paragraph, second paragraph, "The first level, the highest performance level, contains a branch tag, prediction bits, target address, and target instruction bytes for each entry."/ The first level cache is the first level BTB, with the same functionality as a BTB)
- a second level cache configured to store branch prediction information; (Page 409, first paragraph, second paragraph, "The second level, the medium performance level, contains a branch tag (one design eliminates this), prediction bits, and a target address for each entry."/ The second level cache is the second level BTB, with the same functionality as a BTB)

- circuitry coupled to said first level cache and said second level cache (Page 409, fourth paragraph, "Branches in lower performance levels are moved to the highest level on a branch taken execution; this had the benefit that the target instructions can be fetched without increasing memory demands. Entries at higher performance levels are moved to lower performance levels one level at a time as they are replaced.")
- wherein said circuitry is configured to detect said first level cache does not contain branch prediction information corresponding to a first address; (Page 409, first column, first, second, and fourth paragraphs, "...Branches in lower performance levels are moved to the highest level on a branch taken execution;...", page 398, first column, fourth paragraph, "The Branch Target Buffer (BTB) can be used to reduce the performance penalty of branches by predicting the path of the branch and caching information about the branch [30]...As each instruction is fetched from memory, the instruction address is used to index into the BTB. If a valid BTB entry is found for that address, that instruction is a branch"/ The branch prediction information is the cached information about the branch stored in the BTB. The instruction address is the first address. The case exists where the branch target buffer does not have a valid BTB entry for an instruction address. As a result, the case exists where the branch target buffer does not contain branch prediction information corresponding to the instruction address. This scenario is detected by a

multilevel BTB as the prediction information could come from other BTBs, namely lower levels.)

- determine whether said second level cache contains branch prediction information corresponding to said first address (Page 396, second column, third paragraph, "It is also possible to create a multilevel BTB, with different designs at each level, each containing a different subset of these items.", Page 409, first column, fourth paragraph, "Branches in lower performance levels are moved to the highest level on a branch taken execution; ...", Page 398, first column, fourth paragraph, "The Branch Target Buffer (BTB) can be used to reduce the performance penalty of branches by predicting the path of the branch and caching information about the branch [30]...As each instruction is fetched from memory, the instruction address is used to index into the BTB. If a valid BTB entry is found for that address, the instruction is a branch."/ The branch prediction information is the cached information about the branch stored in the BTB. The instruction address is the first address. The case exists where the branch target buffer has a valid BTB entry for an instruction address since a valid BTB entry is "found". It is determined to have the branch prediction information about the instruction address.)
- rebuild a first branch prediction using said information in response to determining said second level cache contains said information, (Page 409, first column, fourth paragraph, "Branches in lower performance levels are moved to the highest level on a branch taken execution; this has the benefit that the target instructions can

be fetched without increasing memory demands.”/The first branch prediction is the branch prediction information stored in the first level BTB. When branches stored the second level BTB have been executed as taken, then they are stored in the first level BTB. This would entail filling in the entry of the first level BTB, or first branch prediction, with the appropriate information.)

- wherein said information comprises a subset of said first branch prediction, (Page 410, See Table XV/The second level BTBs in BTB 6 and 7 contain a subset of information from their respective first level BTBs), and
- store said first branch prediction in a first entry of said first level cache (Page 409, first column, fourth paragraph, “Branches in lower performance levels are moved to the highest level on a branch taken execution; this has the benefit that the target instructions can be fetched without increasing memory demands. / A first entry is an entry of the BTB.)
- wherein said first entry corresponds to said first address. (Page 398, first column fourth paragraph, “As each instruction is fetched from memory, the instruction address is used to index into the BTB.” The branch prediction entries are stored in the first level cache according to their instruction address)

Claim 10:

26. The mechanism of claim 9, wherein said circuitry is further configured to:
- determine if said first entry of said first level cache is available; (Page 398, first column, fourth paragraph, “If a valid BTB entry is found for that address, the instruction is a branch.”/ Therefore, the entry can be invalid and available)

- evict contents of said first entry (Page 409, first column, fourth paragraph, "Entries at higher performance levels are moved to lower performance levels one level at a time as they are replaced.")
- in response to detecting said first entry is not available; (Page 400, column 1, paragraph 2, "An optimization problem exists if a maximum performance BTB is to be designed with a limited number of bits. As both more entries and more information per entry increase performance, the problem is selecting the number of entries and the amount of information per entry that will maximize performance. "Page 403, column 2, fourth paragraph, "The second concept presented in the *BTB Design Optimization* section is, "*When it is necessary to discard a branch from the BTB, discard the branch with the minimum performance potential.* The Least Recently Used (LRU) algorithm is known by experience to work well for replacing (discarding) entries in memory systems, such as main memory pages and cache lines."/ Because of space limitations, the case exists where the entry in the first level cache is not available because it is being used.) and
- store a subset of said contents in said second level cache responsive to said eviction. (Page 409, column 1 and 2, fourth paragraph, "Entries at higher performance levels are moved to lower performance levels one level at a time as they are replaced. " Page 410, See Table XV. BTB 6 and 7)

Claim Rejections - 35 USC § 103

27. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

28. Claims 3,4,11, and 12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Perleberg et al. in *Branch Target Buffer Design and Optimization* and IBM Technical Disclosure Bulletin, *Partial Address Recording in Branch History Tables*.

Claim 3:

29. The method of claim 1,

- wherein said branch prediction corresponds to a first branch instruction
(Perleberg et al., page 398, column 1, fourth paragraph, "If a valid BTB entry is found for that address, the instruction is a branch"), and
- wherein said branch prediction further comprises information indicating a type of said branch instruction.

Perleberg et al. fail to teach that the branch prediction entry contains information indicating the type of branch instruction.

30. IBM teaches a branch history table that stores the partial branch address, partial target address of the branch and a bit that indicates whether the full target address can be represented by concatenating the partial target bits with the most significant program counter (PC) bits (IBM, page 1, "Disclosed are techniques for recording partial address bits in Branch and Target fields in Branch History Table.", page 3, "A special bit per BHT

entry may be used to indicate that the partial TRG address bits cannot fully represent the target.”, page 2, “When the target address is needed for a branch it may be formed simply by concatenating the higher order address bits (0-49) of the current branch with the bits (50-63) recorded at the TRG of the associated BHT entry (if found).”) The entry in the branch history table is the branch prediction. The aforementioned bit is information that indicates the type of branch instruction, whether the branch is PC relative or not.

31. IBM states on page 2, “The basic idea is to record in TRG, for shorter branches, only those bits that the branch and the target may differ.” As a result, fewer bits would be needed to store the target address. On page 1, IBM states, “The benefit is significant reduction of directory circuits, which can be critical for Branch History Tables with large number of entries or for architecture with large number of address bits.” One of ordinary skill in the art would recognize this technique would also reduce the size of the branch target buffer because fewer bits would be needed to store the target address. Given that the number of entries of the branch target buffer remains constant, the reduction in size of entries reduces the size of the branch target buffer. A smaller BTB would optimize the limited space on a processor.

32. It would have been obvious to one of ordinary skill in the art at the time of applicant's invention to add IBM's bit indicating the type of branch (PC relative or not) to the entry of Perleberg et al.'s branch target buffer to create an optimized branch target buffer where only the partial target addresses are stored, as opposed to the full address, in order to reduce the size of the branch target buffer.

Claim 4:

33. The method of claim 3, wherein rebuilding said first branch prediction comprises decoding said branch instruction. (Perleberg et al., page 398, column 1, fourth paragraph, "After the processor finishes executing the branch, it checks to see if the BTB correctly predicted the branch. ...In either case, the branch predication information and branch target address(if changed) must be updated after the branch." Perleberg et al., page 396, column 1, fifth paragraph, "A typical pipeline might have five stages, including instruction fetch, instruction decode, operand fetch, execution, and result writeback." / The branch instruction is decoded, and then executed. The first level BTB is then updated accordingly.)

Claim 11:

34. The mechanism of claim 9,

- wherein said branch prediction corresponds to a first branch instruction
(Perleberg et al., page 398, column 1, fourth paragraph, "If a valid BTB entry is found for that address, the instruction is a branch"), and
- wherein said branch prediction further comprises information indicating a type of said branch instruction

Perleberg et al. fail to teach that the branch prediction entry contains information indicating the type of branch instruction.

35. IBM teaches a branch history table that stores the partial branch address, partial target address of the branch and a bit that indicates whether the full target address can be represented by concatenating the partial target bits with the most significant program

Art Unit: 2183

counter (PC) bits (IBM, page 1, "Disclosed are techniques for recording partial address bits in Branch and Target fields in Branch History Table.", page 3, "A special bit per BHT entry may be used to indicate that the partial TRG address bits cannot fully represent the target." ,page 2, "When the target address is needed for a branch it may be formed simply by concatenating the higher order address bits (0-49) of the current branch with the bits (50-63) recorded at the TRG of the associated BHT entry (if found).") The entry in the branch history table is the branch prediction. The aforementioned bit is information that indicates the type of branch instruction, whether the branch is PC relative or not.

36. IBM states on page 2, "The basic idea is to record in TRG, for shorter branches, only those bits that the branch and the target may differ." As a result, fewer bits would be needed to store the target address. On page 1, IBM states, "The benefit is significant reduction of directory circuits, which can be critical for Branch History Tables with large number of entries or for architecture with large number of address bits." One of ordinary skill in the art would recognize this technique would also reduce the size of the branch target buffer because fewer bits would be needed to store the target address. Given that the number of entries of the branch target buffer remains constant, the reduction in size of entries reduces the size of the branch target buffer. A smaller BTB would optimize the limited space on a processor.

37. It would have been obvious to one of ordinary skill in the art at the time of applicant's invention to add IBM's bit indicating the type of branch (PC relative or not) to the entry of Perleberg et al.'s branch target buffer to create an optimized branch target

buffer where only the partial target addresses are stored, as opposed to the full address, in order to reduce the size of the branch target buffer.

Claim 12:

38. The mechanism of claim 11, wherein rebuilding said first branch prediction comprised decoding said branch instruction. (Page 398, column 1, fourth paragraph, "After the processor finishes executing the branch, it checks to see if the BTB correctly predicted the branch. ...In either case, the branch predication information and branch target address(if changed) must be updated after the branch." Page 396, column 1, fifth paragraph, "A typical pipeline might have five stages, including instruction fetch, instruction decode, operand fetch, execution, and result writeback." / The branch instruction is decoded, and then executed. The first level cache (first level BTB) is then updated accordingly.)

39. Claim 5 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Perleberg et al. in *Branch Target Buffer Design and Optimization* and IBM Technical Disclosure Bulletin, *Partial Address Recording in Branch History Tables*,. as applied to claim 4 above, and further in view of Free On-line Dictionary of Computing (FOLDOC - <http://wombat.doc.ic.ac.uk/foldoc/index.html>).

40. **Note the claim objections above regarding the term "second level cache" **

Claim 5:

41. The method of claim 4, wherein said branch instruction is fetched from said second level cache.

Art Unit: 2183

42. Regarding this matter, Perleberg et al. states, "As each instruction is fetched from memory, the instruction address is used to index into the BTB" (Page 398, fourth paragraph). However, Perleberg et al. is silent on whether the memory can be a second level cache.

43. FOLDOC defines a second level cache as, "A larger, slower cache between the primary cache and main memory. Whereas the primary cache is often on the same integrated circuit as the central processing unit (CPU), a secondary cache is usually external. Furthermore, FOLDOC defines a cache as, "A small fast memory holding recently accessed data, designed to speed up subsequent access to the same data. Most often applied to processor-memory access but also used for a local copy of data accessible over a network etc."

44. Therefore, it would have been obvious at the time of applicant's invention to substitute Perleberg et al.'s memory with the second level cache defined by FOLDOC because a second level cache is a type of memory. It would be advantageous to fetch instructions from a second level cache because it is faster than fetching instructions from main memory.

Claim 13:

45. The mechanism of claim 12, wherein said branch instruction is fetched from said second level cache.

46. Regarding this matter, Perleberg et al. states, "As each instruction is fetched from memory, the instruction address is used to index into the BTB" (Page 398, fourth

paragraph). However, Perleberg et al. is silent on whether the memory can be a second level cache.

47. FOLDLOC defines a second level cache as ,” A larger, slower cache between the primary cache and main memory. Whereas the primary cache is often on the same integrated circuit as the central processing unit (CPU), a secondary cache is usually external. Furthermore, FOLDLOC defines a cache as, “A small fast memory holding recently accessed data, designed to speed up subsequent access to the same data. Most often applied to processor-memory access but also used for a local copy of data accessible over a network etc.”

48. Therefore, it would have been obvious at the time of applicant's invention to substitute Perleberg et al.'s memory with the second level cache defined by FOLDLOC because a second level cache is a type of memory. It would be advantageous to fetch instructions from a second level cache because it is faster than fetching instructions from main memory.

49. Claims 6,7,8,14,15, and 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Perleberg et al. in *Branch Target Buffer Design and Optimization*.

Claim 6:

50. The method of claim 1, wherein said subset comprises a dynamic bit.

51. Perleberg et al. teaches the method of claim 1. However, Perleberg et al. is silent on whether the subset comprises a dynamic bit.

52. By naming the bit “dynamic bit”, the applicant implies the storage of descriptive material in the bits of an entry containing branch prediction information. This difference

is only found in the nonfunctional descriptive material and is not functionally involved in the subset. There is no functionality associated with the dynamic bit in the claims.

53. Therefore, it would have been obvious to a person of ordinary skill in the art at the time of applicant's invention to store any type of data in the subset because such data does not functionally relate to the branch prediction information and because the subjective interpretation of the data does not patentably distinguish the claimed invention.

Claim 7:

54. The method of claim 6, wherein said subset further comprises a branch marker bit.

55. Perleberg et al. teaches the method of claim 6. However, Perleberg et al. is silent on whether the subset further comprises a branch marker bit.

56. By naming the bit "branch marker bit", the applicant implies the storage of descriptive material in the bits of an entry containing branch prediction information. This difference is only found in the nonfunctional descriptive material and is not functionally involved in the subset. There is no functionality associated with the branch marker bit in the claims.

57. Therefore, it would have been obvious to a person of ordinary skill in the art at the time of applicant's invention to store any type of data in the subset because such data does not functionally relate to the branch prediction information and because the subjective interpretation of the data does not patentably distinguish the claimed invention.

Claim 8:

58. The method of claim 7, wherein said branch prediction further comprises an end adjustment bit

59. Perleberg et al. teaches the method of claim 8. However, Perleberg et al. is silent on whether the branch prediction comprises an end adjustment bit.

60. By naming the bit "end adjustment bit", the applicant implies the storage of descriptive material in the bits of an entry containing branch prediction information. This difference is only found in the nonfunctional descriptive material and is not functionally involved in the subset. There is no functionality associated with the end adjustment bit in the claims.

61. Therefore, it would have been obvious to a person of ordinary skill in the art at the time of applicant's invention to store any type of data in the branch prediction because such data does not functionally relate to the branch prediction information and because the subjective interpretation of the data does not patentably distinguish the claimed invention.

Claim 14:

62. The mechanism of claim 9, wherein said subset comprises a dynamic bit.

63. Perleberg et al. teaches the mechanism of claim 9. However, Perleberg et al. is silent on whether the subset comprises a dynamic bit.

64. By naming the bit "dynamic bit", the applicant implies the storage of descriptive material in the bits of an entry containing branch prediction information. This difference

Art Unit: 2183

is only found in the nonfunctional descriptive material and is not functionally involved in the subset. There is no functionality associated with the dynamic bit in the claims.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of applicant's invention to store any type of data in the subset because such data does not functionally relate to the branch prediction information and because the subjective interpretation of the data does not patentably distinguish the claimed invention.

Claim 15:

65. The mechanism of claim 14, wherein said subset further comprises a branch marker bit.

66. Perleberg et al. teaches the mechanism of claim 14. However, Perleberg et al. is silent on whether the subset further comprises a branch marker bit.

67. By naming the bit "branch marker bit", the applicant implies the storage of descriptive material in the bits of an entry containing branch prediction information. This difference is only found in the nonfunctional descriptive material and is not functionally involved in the subset. There is no functionality associated with the branch marker bit in the claims.

68. Therefore, it would have been obvious to a person of ordinary skill in the art at the time of applicant's invention to store any type of data in the subset because such data does not functionally relate to the branch prediction information and because the subjective interpretation of the data does not patentably distinguish the claimed invention.

Claim 16:

69. The mechanism of claim 15, wherein said branch prediction further comprises an end adjustment bit.

70. Perleberg et al. teaches the mechanism of claim 15. However, Perleberg et al. is silent on whether the branch prediction comprises an end adjustment bit.

71. By naming the bit "end adjustment bit", the applicant implies the storage of descriptive material in the bits of an entry containing branch prediction information. This difference is only found in the nonfunctional descriptive material and is not functionally involved in the subset. There is no functionality associated with the end adjustment bit in the claims.

72. Therefore, it would have been obvious to a person of ordinary skill in the art at the time of applicant's invention to store any type of data in the branch prediction because such data does not functionally relate to the branch prediction information and because the subjective interpretation of the data does not patentably distinguish the claimed invention.

73. Claims 17,18,19, and 20 rejected under 35 U.S.C. 103(a) as being unpatentable over Yung (EP 798632 A2) and further in view of Perleberg et al in *Branch Target Buffer Design and Optimization*.

Claim 17:

74. A computer system comprising:

- an interconnect (Yung, figure 1, line connecting element 20 memory with element 22 memory management unit)

- a memory coupled to said interconnect (Yung, figure 1, element 20 memory connected to element 22 via interconnect)
- a second level cache configured to store branch prediction information; (Yung, figure 3/ column 4, lines 30 – 40)
- a processor including a first level cache (Yung, figure 4, element 50 Ultra Sparc Processor, figure 5, element 50 includes L1 cache, column 5, lines 8 – 9), wherein said processor is configured to:
 - detect said first level cache does not contain branch prediction information corresponding to a first address (Yung, column 1, lines 22 – 25, column. 4, lines 13 – 15, lines 25 – 28/ The branch prediction information is the L1 cache entry (Yung, figure 2). The first address is the current address (Yung, column 1, line 24 – 25). A match with the address tag indicates whether there is an entry that contains branch prediction information corresponding to the current address.)
 - determine whether said second level cache contains branch prediction information corresponding to said first address, (Yung, column 1, lines 11 – 16, “If the needed instruction or data is not in this cache, access is then made to a second cache, which is typically external and made with SRAM chips which are faster than the normal DRAM used in main memory.” lines 22 – 25, column 3, lines 30 – 40)
 - rebuild a first branch prediction using said information in response to determining said second level cache contains said information (Yung, column 4, lines 52 – 55, “The second level cache may also store the partially decoded instruction

class, thus eliminating the need to redecode this upon writing this information back into the first level cache on the microprocessor”)

- wherein said information comprises a subset of said first branch prediction, and
- store said first branch prediction in a first entry of said first level cache (Yung, column 4, lines 52 – 55, column 1, lines 19 – 22)

However, Yung fails to teach that second level cache contains information, which comprises a subset of said first branch prediction.

75. Perleberg et al. in *Branch Target Buffer Design and Optimization* teaches that a second level BTB (second level cache) can contain a subset of information stored in the first level BTB (first level cache). (Perleberg et al., page 410, Table XV, BTB 6 and 7)

76. Substituting Perleberg et al.’s design of second level BTB containing a subset of first level BTB branch prediction information into Yung’s multilevel cache would optimize the use of limited bits by having different performance levels for predictions (Perleberg et al. page 409, second paragraph) and provide a performance gain dependant on the implementation of the multilevel cache structure and type of processor. (Perleberg et al., page 409 – 410, third paragraph starting with 1) Results of Simulating Multilevel BTB’s up to IX. Summary and Conclusions). It would have been obvious to one of ordinary skill in the art at the time of applicant’s invention to substitute Perleberg et al.’s design of second level BTB containing a subset of first level BTB branch prediction information into Yung’s multilevel cache to optimize the use of limited bits.

Claim 18:

77. The system of claim 17,

- wherein said processor is further configured to determine if said first entry of said first level cache is available (Yung, figure 4, element 50 Ultra Sparc Processor, figure 5, element 50, element 74/ column 1, lines 9 – 14)
- evict contents of said first entry in response to detecting said first entry is not available; (column 4, lines 25 – 29, “When a cache line is victimized, upon a context switch, address conflict, or otherwise, the cache entry will be lost, along with its branch prediction information. According to the present invention, this information is then stored in the L2 cache”. column 2, lines 28 – 31) and
- store a subset of said contents in said second level cache responsive to said eviction (column 4, lines 25 – 29/ Yung stores the branch prediction information from the L1 cache into the L2 cache)

However, Yung is silent about storing just a subset of branch prediction information.

78. Perleberg et al. in *Branch Target Buffer Design and Optimization* teaches that a first level BTB (first level cache) can store a subset of its branch prediction information into the second level BTB (second level cache) (Perleberg et al., page 409, first column, fourth paragraph, “Entries at higher performance levels are moved to lower performance levels one level at a time as they are replaced.” page 410, Table XV)

79. Substituting Perleberg et al.’s design of storing a subset of branch prediction information in the second level branch target buffer into Yung’s multilevel cache would optimize the use of limited bits by having different performance levels for predictions (Perleberg et al. page 409, second paragraph) and provide a performance gain dependant on the implementation of the multilevel cache structure and type of

processor. (Perleberg et al., page 409 – 410, third paragraph starting with 1) Results of Simulating Multilevel BTB's up to IX. Summary and Conclusions). It would have been obvious to one of ordinary skill in the art at the time of applicant's invention to substitute Perleberg et al.'s design of storing a subset of branch prediction information in the second level BTB to Yung's multilevel cache structure to optimize the use of the bits.

Claim 19:

80. The system of claim 17,

- wherein said branch prediction (Yung, figure 2, column 4, lines 12 – 29/ the fields of the L1 cache entry comprise a branch prediction when the instruction is a branch)
- corresponds to a first branch instruction, (Yung, column 4, lines 14 – 18/ The first branch instruction has an L1 cache entry with the instruction class field set to branch.), and
- wherein said branch prediction further comprises information indicating a type of said branch instruction (Yung, column 4, lines 15 – 18)

Claim 20:

81. The system of claim 19,

- wherein rebuilding said first branch prediction comprises decoding said branch instruction. (Yung, column 5, lines 1 –3/ The decode occurs once when the entry initially enters the multilevel cache structure. The rebuilding process benefits from the initial decoding.)

Conclusion

82. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

83. The following illustrate different embodiments of the two level branch prediction cache. The second level cache has limited prediction information compared to the first level cache.

- Stiles et al. (US Pat. 6,606,616) *Branch Prediction Device with Two Levels of Branch Prediction Cache*
- Stiles et al. (US Pat. 5,515,518) *Two-Level Branch Prediction Cache*
- Stiles et al. (US Pat. 5,163,140) *Two-Level Branch Prediction Cache*

84. Another embodiment of a two level branch prediction table. The following stores a hint in software to choose whether to use the prediction stored in the hint, or what is already stored in the branch prediction table.

- Yeh, et al. (US Pat. 6,553,488) *Method and Apparatus for Branch Prediction Using First and Second Level Branch Prediction Tables*

85. Another embodiment of a two level prediction storage.

- Blaner et al. (US Pat. 5,423,011) *Apparatus for Initializing Branch Prediction Information*

86. Stores whether branch is a call in branch history table

- Emma et al., (US Pat. 5,276,882) *Subroutine Return Through Branch History Table*

87. Similar to branch marker bits.

Art Unit: 2183

- Tran (US Pat. 6,141,748) *Branch Selectors Associated with Byte Ranges Within an Instruction Cache for Rapidly Identifying Branch Predictions*

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Latha Ravindran whose telephone number is (703)305-8115. The examiner can normally be reached on Monday through Friday 8:30am to 5:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703) 305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).


Latha Ravindran
Examiner
Art Unit 2183


EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Application/Control Number: 09/912,011
Art Unit: 2183

Page 31